Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung AW2

Eike Falkenberg

Sicherheits Aspekte im Pervasive Gaming Framework

Fakultät Technik und Informatik
Studiendepartment Informatik

Eike Falkenberg

Thema der Ausarbeitung AW2
Sicherheits Aspekte im Pervasive Gaming Framework
Stichworte
Pervasive Gaming Sicherheit
Kurzzusammenfassung
In unserem Master Projekt haben wir ein Spiele Framework für pervasive Spiele entwickelt. Die in dem Kontext des Projekts relevanten Sicherheits Aspekte werde ich in dieser Ausarbeitung beschreiben.

Inhaltsverzeichnis

1 Einleitung

In unserem Master Projekt haben wir ein Framework für pervasive Spiele entwickelt. Dieses Framework soll als Basis für die Entwicklung von mobilen Spielen, die den realen Kontext einbinden, dienen.
Ich habe mich im Rahmen dieses Projektes mit dem Thema Sicherheit beschäftigt. In meinem Vortrag habe ich mich hauptsächlich um die Absicherung der Anwendung auf den Clients selber konzentriert. In dieser Ausarbeitung will ich die Sicherheit der Gesamtarchitektur beschreiben, somit kommt ein weiterer Aspekt hinzu: die Kommunikation zwischen dem Client und dem Server.

2 PGF - Das Pervasive Gaming Framework

2.1 Überblick

Jan Napitupulu schrieb 2005 an der HAW eine Bachelor Arbeit mit dem Titel „Pervasive Gaming. Entwicklung von ortsabhängiger Spielesoftware“ (Nap06). Er beschreibt in seiner Arbeit die Idee einer Handels Simulation als pervasives Spiel. Inspiriert von dieser Bachelorarbeit haben wir in unserem Master Projekt ein Framework für pervasive Spiele entwickelt. Unser Framework soll als Grundlage der Realisierung für Spielideen wie der von Jan Napitupulu dienen.
Wir haben unser Framework auf Basis des Microsoft .NET Compact Framework (Whe03) entwickelt. Es handelt sich hierbei um die abgespeckte Version von Microsoft .NET für mobile Endgeräte.
Unser Framework soll den Entwickler pervasiver Spiele von der darunterliegenden Technik abstrahieren lassen und so die Entwicklung solcher Applikationen beschleunigen. Positionierungstechnologie, Kommunikationstechnik und die Synchronisierung sind im Kern die Techniken, die wir kapseln wollen.
Zu unserem Framework gehört auch eine prototypische Anwendung, ein Schnitzeljagd Spiel. Dieses Spiel sollte uns zum einen dienen, die Funktionalität des Frameworks zu testen, dient aber auch dem potentiellen Entwickler als Grundlage für eigene Spiele oder zumindest als Beispiel wie das Framework verwendet werden kann. Da das Hauptaugenmerk nicht auf dem Spiel, sondern auf dem Framework lag, ist das Spiel sehr einfach gehalten. Positionsabhängig bekommt ein Spieler Fragen, muss diese beantworten und geht dann weiter zu der nächsten Position. Unser Spiel besteht wie auch das Framework selbst, aus einer Client und einer Server Software.
Als Hardware für die Clients standen uns HP iPAQ hw6915 mit Microsoft Windows Mobile 5 als Betriebssystem zur Verfügung. Da diese Geräte auch über einen integrierten GPS Emp-

fänger verfügen, waren sie für unsere Zwecke sehr gut geeignet. Da die Hardware es anbot, nutzt unser Prototyp GPS als Positionierungssystem.

2.2 Grundlegende Funktionsweise

Ich werde hier nur auf die wichtigsten Punkte eingehen, die zum Verständnis der Sicherheitskonzepte notwendig sind. Eine ausführliche Beschreibung der Konzepte und Architektur befindet sich in dem Projektbericht.
Das PGF besteht aus einer Server und einer Client Komponente. Pro Spieltyp existiert eine Instanz der Server Software und n Instanzen der Client Software. Pro Spiel gibt es einen Spielführer. Der Spielführer erzeugt über eine Weboberfläche auf dem Server ein neues Spiel. Er gibt dort an wie das Spiel heist, wann es beginnt und wählt die Route aus. Danach können sich Spieler für dieses Spiel registrieren und bekommen vom Server eine Route zugewiesen. Die gesamte Kommunikation zwischen Client und Server findet über Web Services statt. Die Sicherung dieser Kommunikation bildet einen Teil des Sicherheitskonzeptes. Aufgrund der derzeitigen Situation können wir nicht davon ausgehen, dass die Clients, also die mobilen Endgeräte, permanent online sind. Wir haben uns daher für ein Datenhaltungskonzept entschieden, in dem die Clients am Anfang eines Spieles alle Daten herunterladen und am Ende ihre Ergebnise auf den Server hochladen. Das zwischenzeitliche Synchronisieren während des Spieles ist zwar vorgesehen, bleibt jedoch optional. Daraus resultiert, dass die Daten auf dem Client vorgehalten werden müssen. Die Daten auf den Clients sollen dem Benutzer nicht zugänglich sein und die Integrität der Daten muss gewährleistet werden. Dies ist der zweite des Sicherheitskonzeptes.
Das Bundesamt für Sicherheit in der Informationstechnik gibt eine Broschüre (bsi) heraus, die bei der Entwicklung sicherer Infrastrukturen im mobilen Bereich dient. Das PGF ist ein eher untypisches, mobiles System, da die Clients nicht in eine gemeinsame Infrastruktur integriert sind. Ferner hat jeder Spieler seinen eigenen Client. Trotzdem hat die Broschüre einige Sicherheitsprobleme offenbart, die wir bei der Entwicklung des PGF berücksichtigt haben, zum Beispiel bei der Verwendung der Bluetooth Technologie.
Ich werde mich in dieser Ausarbeitung auf die beiden genannten Sicherheits Aspekte beschränken, weitere Risiken und warum diese nicht weiter betrachtet wurden, werden in der Konklusion erläutert.

3 Integrität der Daten auf dem Client

3.1 Integrität der Daten

Wie bereits in 2.2 geschildert sind wir dazu gezwungen Daten des Spieles auf den Clients abzulegen. Daraus ergibt sich das Problem, dass auf dem Gerät des Benutzers Daten vor-

liegen, die für ihn unter Umständen noch gar nicht sichtbar sein sollen. In unserem prototypischen Spiel könnte dies die Route eines Spieles sein. Kennt ein Spieler bereits zu Beginn die komplette Route hat er so einen Vorteil gegenüber den anderen Spielern.
Typischerweise würde man diese Daten verschlüsselt ablegen, jedoch braucht man zum entschlüsseln einen Schlüssel, der ebenfalls auf dem Client abgelegt werden muss. Es ist so rasch absehbar, dass dieses Problem mit derzeitigen Möglichkeiten nicht lösbar ist. So lange die Daten auf dem Client autonom entschlüsselt werden müssen, sind die Daten nicht effizient zu verschlüsseln. Der richtige Ansatz wäre, die Daten gar nicht erst beim Benutzer abzulegen oder den Schlüssel zum entschlüsseln erst bei Bedarf zu übertragen, leider widerspricht dies den Anforderungen.
In der nächsten Entwicklungsstufe des PGF wird die Ad-hoc Übertragung des Schlüssels zum entschlüsseln der Daten jedoch der einzig logische Schritt sein. Somit erhält der Nutzer alle Daten verschlüsselt vorab, sichtbar werden die Informationen jedoch erst, wenn er an der richtigen Position ist.

3.2 Integrität der Software

Ein weiteres Problem besteht darin, dass die Software selbst manipuliert werden kann. Denkbar ist eine manipulierte Software die vorgibt, dass der Client an der Zielposition ist, obwohl er sich an einer anderen befindet. Bei Desktop Computer Spielen wird dieses Problem mit einer Zusatzsoftware bekämpft, die in kurzen Intervallen prüft, ob die ausführbare Datei des Programms verändert wurde. Aufgrund unserer Architektur und der darunter liegenden Plattform lässt sich dieses Problem kaum in den Griff bekommen. Wenn wir eine zusätzliche Software einsetzen würden, die den Zustand der ausführbaren Datei überprüft, so könnte dieses zusätzliche Programm ebenfalls kompromitiert werden und wäre somit wirkungslos. Desktop Sicherungsprogramme versuchen dieses Problem mit dem permanentem Austausch von Hashes über das Internet zu beheben. Da wir aber keine permanente Internetverbindung haben und die Performance der mobilen Endgeräte keine zusätzlichen Prozesse zulässt, werden wir eine solche Software nicht nutzen können. Dieses Problem lässt sich so nicht effizient lösen.

3.3 Trusted Computing

Wie bereits aufgeführt, sind softwarebasierte Lösungsansätze leicht auszuhebeln und bieten nicht den erforderlichen Schutz. Ein bereits in der Präsentation vorgestellter Ansatz ist daher, die Sicherheitsfunktionen Hardware basiert anzubieten.
Die Trusted Computing Group (TCG) als Nachfolgeorganisation der wenig populären Trusted Computing Platform Alliance(TCPA) entwickelt Standards zu hardwarebasierten Sicherheitsfunktionen (tcgb). Es exisitiert eine eigene Gruppe für mobile Endgeräte, die entsprechende

Spezifikationen entwickelt und verabschiedet (tcga).
Trust will die TCG dabei wie folgt verstanden wissen:

> Trust is the expectation that a device will behave in a particular manner for a specific purpose. A trusted platform should provide at least three basic features: protected capabilities (resources), integrity measurement and integrity reporting. *[Trusted Computing Group]*

Die TCG beschreibt damit genau die Anforderungen, die ich definiert habe: Geschützte Bereiche auf dem System und die Sicherstellung der Integrität von Daten (ausführbare Dateien sind auch Daten).

3.3.1 Trusted Platform Module

Die ersten TCG Spezifikationen beschäftigten sich nur mit der Sicherheit von PCs und Servern, inzwischen existieren auch Spezifikationen für Smartphones und PDAs.Kernelement der TCG-Spezifikation ist das Trusted Platform Module (TPM), welches unter anderem folgende Funktionalitäten bietet:

- Integritätsüberprüfung von Software und Hardware
- Sicheres Erzeugen, Verwalten und Bereitstellen von Schlüsselmaterial für Sicherheitsapplikationen
- Sicheres Booten mit Überwachung der einzelnen Bootschritte
- Optionale weitere Schritte wie Festplattenentschlüsselung

Das TPM als Hardware Chip bietet die Möglichkeit, Schlüssel sicher zu erzeugen und zu verwahren. Die wichtigsten Sicherheits Algorithmen sind auf dem Chip vorhanden, auch ein für Crypto Anwendungen unerlässlicher, physikalischer Zufallszahlen Generator ist auf dem Chip vorgesehen (s. Abb. 1).

Die von der TCG verabschiedete Spezifikation deckt somit alles ab, was wir benötigen. Leider unterstützt noch kein Betriebssystem die derzeit verfügbaren TPMs. Es existiert lediglich Linux basierte, experimentelle Betriebssysteme, die die angebotenen Sicherheitsfunktionen unterstützen. Somit ist TPM eine Option, die wir leider nicht wählen konnten. Eine Integration von TPM wäre sicher eine interessante Aufgabe als Master Arbeit, den Rahmen des Projektes hat es aber leider gesprengt. Auch sind mobile Endgeräte mit den Crypto-Chips derzeit noch nicht verfügbar. Somit bleibt zumindest zu sagen, dass bei künftiger Verfügbarkeit von TPM und entsprechender Betriebssystemunterstützung der Einsatz sehr zu empfehlen wäre.

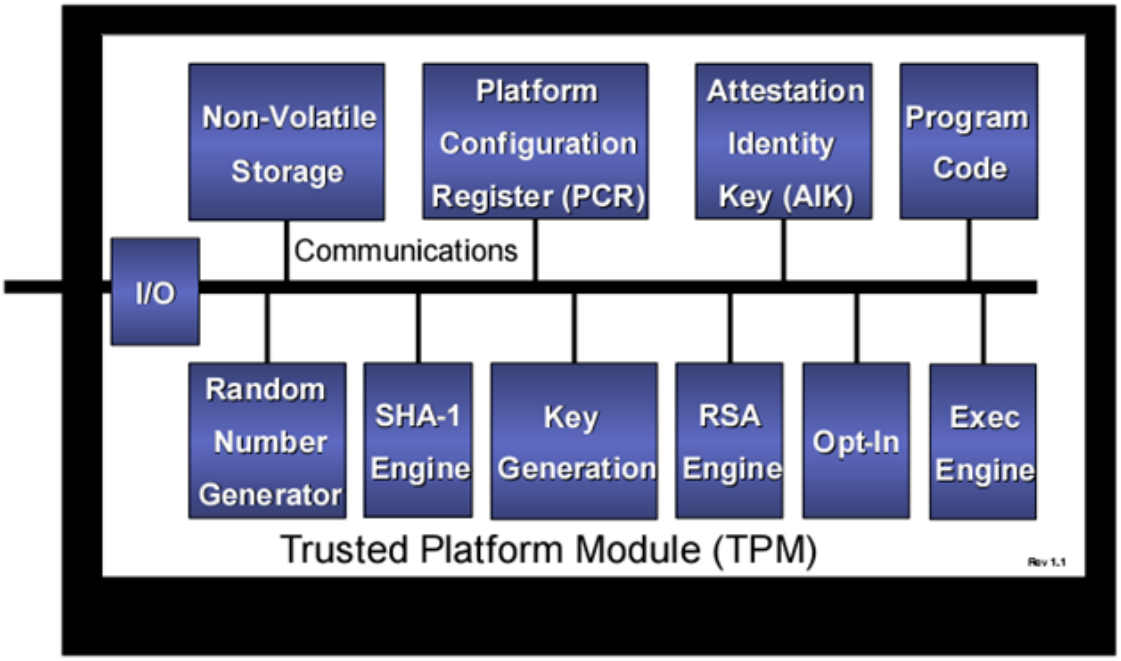

Figure 4:g – TPM Component Architecture

Abbildung 1: TrustedPlatformModule ComponentArchitektur

4 Sicherheit in der Client Server Kommunikation

Wie bereits zuvor dargestellt, findet die gesamte Kommunikation im PGF auf Basis der Web Services Technologie statt.
Wie so oft wurde das Thema Sicherheit auch bei der Entwicklung der Web Services erst im nachhinein berücksichtigt und dann auf die bestehende Technik aufgesetzt. Erste Web Services integrierten einfach Benutzername und Passwort als zusätzliche Parameter in ihre Aufrufe. Schnell wurde klar, dass dies weder elegant, noch flexibel und vor allem nicht einheitlich ist. Gefragt waren Standards, die einheitliche, sichere und dabei flexible Zugriffe auf Web Services ermöglichen.
Die meisten Web Services Implementierungen basieren auf speziellen XML-Nachrichten im SOAP Format, welche in einen Header- und einen Body Bereich aufgeteilt sind. Im Header werden Meta-Informationen abgelegt, im Body die eigentlichen Nutzdaten. Für eine einheitliche Sicherheitsintegration in Web Services ist die Definition von zusätzlichen Header Informationen der logische Schluss. Genau das ist es, was die Organization for the Advancement of Structured Information Standards (OASIS) getan hat; sie hat in diversen Standards solche Header beschrieben und diese verabschiedet.

Heute ist es so weit, dass Teile dieser Spezifikationen bereits in Produkten erhältlich sind und daher auch für unser Projekt nutzbar. Ich werde im folgenden einige dieser Standards vorstellen, um anschliessend darauf einzugehen, ob und wie diese in unserem Projektkontext von nutzen sein können.

4.1 Web Services Security

Als Web Services Security oder auch WS-Security oder nur WSS wird ein von der OASIS verabschiedeter Standard für die Zugriffsauthentifizierung auf Web Services und die Integrität der dabei ausgetauschten Nachrichriten bezeichnet. Die derzeit aktuelle Fassung 1.1 liegt seit Februar 2006 vor und ist öffentlich verfügbar (oasa). Web Services Security bildet die Basis für weitere Spezifikationen. Web Services Security definiert ein zusätzliches Element *wsse:Security* mit weiteren Subelementen im SOAP Header für die Zugriffsauthentifizierung und beschreibt das Signieren und Verschlüsseln von SOAP Nachrichten.

4.1.1 User Name Token

User Name Token ist die einfachste Art, wie ein Requestor sich bei einem Web Service Authentifizieren kann. User Name Token besteht aus zusätzlichen SOAP Header Elementen, die beim Aufruf mit übergeben werden und einen Benutzernamen und ein, optional gehashtes, Passwort beinhalten.

```
<wsse:Security>
    <wsse:UsernameToken wsu:Id="...">
        <wsse:Username>Eike</wsse:Username>
        <wsse:Password>MeinPasswort</wsse:Password>
    </wsse:UsernameToken>
</wsse:Security>
```

4.1.2 Binary Security Token

In größeren IT Infrastrukturen wird häufig der verteilte Authorisierungsdienst Kerberos oder eine Public Key Infrastructure (PKI) eingesetzt. In so einem Umfeld langt ein User Name Token nicht aus. Für diesen Fall existiert das Binary Security Token. Es definiert die Beschreibung und die Einbettung des Tickets oder Zertifikats, das zur Authentifikation verwendet werden soll.
Ein Beispiel für ein eingebettetes Kerberos Ticket:

```
<wsse:Security>
    <wsse:BinarySecurityToken
```

```
            ValueType="wsse:Kerberosv5ST"
            EncodingType="wsse:Base64Binary">
    ZAB69RT7UK ...
    </wsse:BinarySecurityToken>
</wsse:Security>
```

Das Pendant für eine PKI mit x.509 Zertifikaten:

```
<wsse:Security>
    <wsse:BinarySecurityToken
      ValueType="wsse:X509v3"
      EncodingType="wsse:Base64Binary">
    ABCdef123 ...
    </wsse:BinarySecurityToken>
</wsse:Security>
```

4.1.3 Signaturen und Verschlüsselungen

Da sich XML-Encryption und XML-Signature bewährt haben, beschreibt WSS die Einbindung in SOAP Nachrichten. Für ein Element mit der id *SignedPart* könnte der Header für eine SHA-1 Signatur so aussehen:

```
<ds:Signature>
  <ds:SignedInfo>
    <!-- [..] -->
    <ds:Reference URI="#SignedPart">
      <ds:Transforms>
        <ds:Transform
         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      </ds:Transforms>
      <ds:DigestMethod
        Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>ZYXwvu987...</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>ABCdef123....</ds:SignatureValue>
  <!-- [..] -->
</ds:Signature>
```

Die Verwendung von XML-Encryption ist entsprechend.

4.2 WS-*

Diverse Standards im Kontext der Web Services Technologie werden unter dem Kürzel WS-* zusammengefasst. Einige dieser Standards sind weiterführende Standards, die auf WSS aufbauen. Diese Standards bieten Flexibilität und Integrationsmöglichkeiten, die mit WSS individuell konzipiert werden müssten und so nicht einheitlich würden. Diese Standards sind:

- WS-SecurityPolicy
- WS-Trust
- WS-SecureConversation
- WS-Federation

4.2.1 WS-SecurityPolicy

Das bisher vorgestellte Konzept von WSS hat ein gravierendes Problem: der Requestor muss erfahren, welche Security Tokens der Service akzeptiert. Dies könnte dem Entwickler überlassen werden. Problematisch wird es jedoch, wenn der Service die akzeptierten Tokens zu einem späteren Zeitpunkt ändert. Der Service kennt nicht alle seine Requestor und kann ihnen die Veränderung somit nicht mitteilen. Genau dort setzt WS-SecurityPolicy, oder kurz WS-Policy, an. WS-Policy definiert eine einheitliche Syntax, um deklarativ die Sicherheitsanforderungen eines Web Services zu beschreiben. Der Requestor kann so vor dem Aufruf des Web Services prüfen, ob er die geforderten Credentials erfüllen kann und flexibel auf die Anforderungen reagieren.

```
<wsp:Policy>
  <sp:SymmetricBinding>
    <wsp:Policy>
      <sp:ProtectionToken>
        <wsp:Policy>
          <sp:KerberosV5APREQToken
            sp:IncludeToken=".../IncludeToken/Once" />
        </wsp:Policy>
      </sp:ProtectionToken>
      <sp:SignBeforeEncrypting />
      <sp:EncryptSignature />
    </wsp:Policy>
  </sp:SymmetricBinding>
  <sp:SignedParts>
```

```
    <sp:Body/>
    <sp:Header
       Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    />
  </sp:SignedParts>
  <sp:EncryptedParts>
    <sp:Body/>
  </sp:EncryptedParts>
</wsp:Policy>
```

Dies ist ein typisches Beispiel solch einer Policy aus der Spezifikation (oasc).
In diesem Beispiel wird ein Kerberos Ticket erwartet, die Nachricht muss vor dem Verschlüsseln signiert werden und die Signatur muss mitverschlüsselt werden. Das komplette Body Element, sowie alle Elemente aus dem Header die zum dem Addressing Namensraum gehören müssen signiert werden. Verschlüsselt wird der Body der Nachricht.

4.2.2 WS-Trust

Durch WS-Policy erhält der Service die Möglichkeit, seine Anforderungen dynamisch dem Requestor mitzteilen. Dadurch ergbit sich jedoch ein neues Problem: wenn der Requestor die Credentials nicht erfüllen kann, ist es ihm nicht möglich den Service aufzurufen.
Es wird ein Instanz in der Infrastruktur benötigt, die dem Requestor bescheinigen kann, dass er berechtigt ist den Service aufzurufen, auch wenn er nicht die notwendigen Credentials liefern kann.
An diesem Punkt setzt WS-Trust (oasd) an und führt eine zentrale Stelle zum erzeugen, prüfen und entwerten von Security Tokens ein, den Security Token Service (STS). Der STS ist dem Service bekannt und somit kann der Service Security Tokens vertrauen, die von dem STS ausgestellt wurden. So kann ein Requestor, der nur ein x.509 Zertifikat liefern kann, auch einen Service aufrufen, der ein Security Assertion Markup Language (SAML) Token erwartet. Die Kommunikation zwischen Requestor und dem STS wird über Request Security Token (RST) und Request Security Token Response (RSTR) definiert.
Die Abbildung 2 stellt grafisch dar, wie der Requestor zuerst den STS aufruft und anschließend mit dem erhaltenen SAML-Token direkt den Service aufruft.

4.2.3 WS-SecureConversation

WS-SecureConversation (oasb) ist eine Spezifikation, die WS-Trust optimiert. Bei WS-Trust kann der STS stark belastet werden, da bei jedem Aufruf der Requestor erneut den STS aufruft und erst anschließend den eigentlichen Service. WS-SecureConversation beschreibt, wie Requestor und Service einen gemeinsamen Sicherheitskontext aufbauen können und

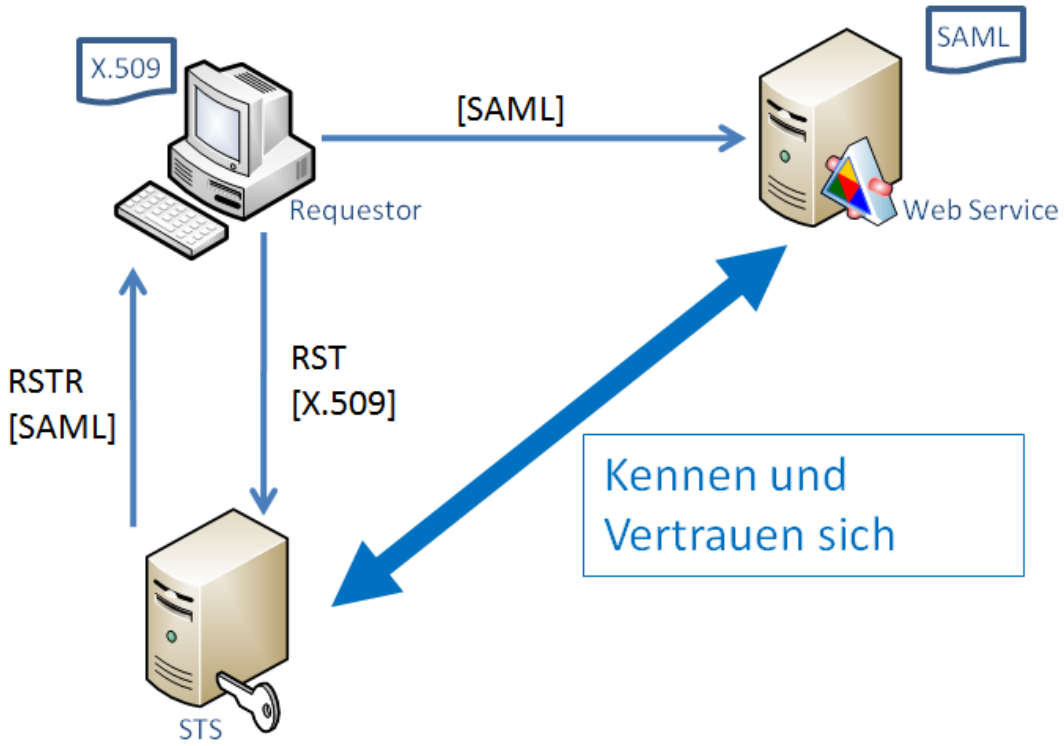

Abbildung 2: WS-Trust

der Requestor so mehrere Anfragen hintereinander direkt an den Service starten kann. Begonnen wird wie bei WS-Trust unter Einbeziehung des STS, dann wird ein Sitzungsschlüssel zwischen Requestor und Service vereinbart, von diesem leiten sich dann beide Folgeschlüssel ab.

4.2.4 WS-Federation

Befinden sich Requestor und Service in demselben Realm, ist es ohne weiteres möglich, die Authorisierung für eine Identität zu prüfen. Schwierig wird es jedoch, wenn der Service sich in einem anderen Realm befindet, wie der Requestor. Das von dem STS ausgestellte SAML-Token kann der Service nicht als vertrauenswürdig einstufen, da er den STS nicht kennt und ihm somit nicht vertrauen kann.
Dieser Problematik nimmt sich WS-Federation (fed) an. WS-Federation beschreibt, wie sich STS aus unterschiedlichen Realms vertraut machen können und so ein Client realmübergreifend für einen Service Zugriff authorisiert werden kann. In WS-Federation ruft der Requestor zuerst seinen STS auf und holt fordert ein SAML-Token an. Mit dem erhaltenen SAML Token

ruft der Requestor den STS des fremden Realms auf und bekommt von diesem ein SAML Token für den Service im fremden Realm.

4.3 Einbettung in das PGF

Da wir sowohl den Server, als auch den Client auf dem Microsoft .NET Framework aufgebaut haben, können wir die frei verfügbaren Web Services Enhancements (WSE) nutzen. WSE integriert viele WS-* Standards und auch WS-Security in die .NET Web Services.
Für unsere Zwecke langt ein User Name Token mit gehashten Passworten. Denkbar wäre auch eine PKI Unterstützung, auch das wäre mit den WSE kein Problem. Für unsere bisherigen Zwecke sind WS-Policy und auch WS-Trust kaum sinnvoll nutzbar. Wird das PGF allerdings im größeren Umfeld eingesetzt, könnte vor allem eine Integrierung von WS-Policy sinnvoll werden, um mehr Flexibilität zu erreichen.

5 Konklusion

Ich habe in dieser Ausarbeitung und auch in der Sicherheitsbetrachtung im Projekt bewusst den Fokus auf die beiden dargestellten Sicherheits Aspekte gelegt. Diebstahl des Gerätes, Viren auf den Geräten, Man-in-the-Middle Angriffe und vieles mehr hätten auch berücksichtigt werden können. Leider ließ es der Rahmen des Projektes nicht zu, so tief in die Problematik der Absicherung von mobilen Anwendungen einzusteigen.
Zusammenfassend kann man sagen, dass die Integrität des Clients 3 das größte Problem ist. Ohne massiven Aufwand oder der Nutzung von Hardwarechips ist die Integrität nicht sicherzustellen. Massiver Aufwand bedeutet auch Performance Einbußen, was auf den ohnehin schon schwachen Geräten nicht akzeptabel ist. Die zusätzlichen Hardware Chips werden sicher in den nächsten Jahren auch auf mobilen Geräten verfügbar sein und von den Betriebssystemen unterstützt werden, derzeit aber leider noch nicht.
Die Sicherheit in der Client Server Kommunikation 4 ist im Gegensatz zu der Client Integrität ein weitgehend gelöstes Problem. Es existieren akzeptierte Standards, die eine solide und denoch flexible Absicherung der Kommunikation sicherstellen. Wie bereits in 4.3 erläutert, macht der Gebrauch von User Name Tokens im Kontext des PGF Sinn, die zukünftige Integration weiterer Standards wie WS-Policy muss abgewogen werden.

Literatur

[bsi] *Bundesamt für Sicherheit in der Informationstechik, Mobile Endgeräte und mobile Applikationen: Sicherheitsgefährdungen und Schutzmaßnahmen*. http://www.bsi.bund.de/literat/doc/mobile/mobile_endgeraete.pdf

[fed] *WS-Federation 1.1*. http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-fed/WS-Federation-V1-1B.pdf

[Nap06] NAPITUPULU, Jan: *Pervasive Gaming. Entwicklung von ortsabhängiger Spielesoftware*. Vdm Verlag Dr. Müller, 2006

[oasa] *OASIS Standard, Web Services Security v1.1*. http://www.oasis-open.org/specs/index.php#wssv1.1

[oasb] *WS-SecureConversation 1.3 (Committee Specification 01)*. http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-spec-cs-01.pdf

[oasc] *WS-SecurityPolicy 1.2 (Committee Draft 01)*. http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512/ws-securitypolicy-1.2-spec-cd-01.pdf

[oasd] *WS-Trust 1.3 (Committee Specification 01)*. http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-spec-cs-01.pdf

[tcga] *Trusted Computing Group, Mobile Phone Specifications*. https://www.trustedcomputinggroup.org/specs/mobilephone

[tcgb] *Trusted Computing Group, Trusted Platform Module (TPM) Specifications*. https://www.trustedcomputinggroup.org/specs/TPM

[Whe03] WHEELRIGHT, Robert: *Microsoft . NET Compact Framework. Core Reference*. Microsoft Press, 2003